What to Eat in Kyoto

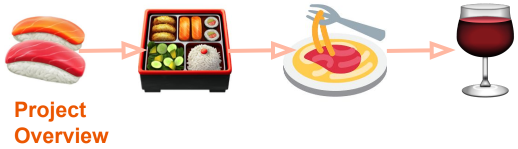
By Alisa Babikova, Yingkun (Will) Zhu, WanQi(Angie) Tay

Optimization and Simulation Methods for Analytics

August 21st, 2018

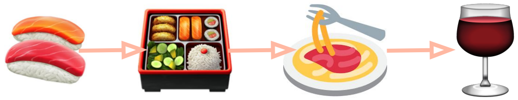
Agenda

- Project Overview
  - Problem Statement
  - Dataset Overview and Context
  - Data Processing
  - Data Structure
- Optimization
  - Models in 3 languages Overview
  - Individual Optimization Model
  - Combined Optimization Model
  - Challenges
- Visualizations and Recommendations
  - Planning your trip
  - Maps
- Conclusion and Future Work



Problem Statement

- Japan is seeing a record-breaking amount of visitors every year for the past few years, and Kyoto is not an exception.
- Picking the place to eat given constraints of your trip is a frustrating task - especially if you aren't familiar with culture.
- We used data from a crowd-sourced restaurant-rating service Tabelog with detailed information on 800+ restaurants in Kyoto to simplify the process of picking the places to eat.
- By optimizing the process of choosing a place to eat that maximizes the rating, we hope to optimize visitors' travelling plans as well as encourage the restaurants to keep up the good work to stay in our recommendations.



Dataset Overview and Context

- Source: [Kaggle]
- Size: 13 variables and 800+ observations
- Original Variables:
  - "Name" "JapaneseName" "Station"
  - "FirstCategory" "SecondCategory""DinnerPrice" "LunchPrice"
  - "TotalRating" "DinnerRating" "LunchRating" "ReviewNum"
  - "Lat" "Long"

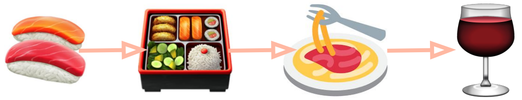
```
head(raw.data)
```
```

| | X <int> | Name <fctr> | JapaneseName <fctr> | Station <fctr> |
|---|---|---|---|---|
| 1 | 1 | Orudeidainingurajou | オールデイダイニング　ラジョウ | Kyoto |
| 2 | 2 | Steak Frites Gaspard zinzin | ステックフリット ガスパール ザ... | Karasuma |
| 3 | 3 | KAZUMA | 和馬 | Sanjo |
| 4 | 4 | okonomiyakiteppanyakimiki | お好み焼き 鉄板焼き 三喜 | Tambaguchi |
| 5 | 5 | Shaofeiyan | 小肥羊 京都河原町店 | Kyoto Shiyakusho Mae |
| 6 | 6 | okuta-va | OCTAVAR | Kyoto |
```

Data Processing

- Category-wise:
  - Merged the FirstCategory with the SecondCategory into Category
  - Applied OpenRefine to clean up a few additional categories and kept the main cuisines
- Price:
  - Original dataset contained price range in JPY
  - Split the upper and lower bound for Dinner minimum, Dinner maximum
  - Calculated the AverageDinnerPrice for further analysis
  - Converted the AverageDinnerPrice into USD (AverageDinnerPriceInUSD) for better understanding
- Lat/Long:
  - To project the recommended restaurants onto Google Map more accurately, we transformed restaurants' original location (in Geographic coordinate system) onto 2D coordinate system - Universal Transverse Mercator (UTM).

julia



Project Overview

Data Processing - Continued

```r
```{r}
raw.data <- read.csv("Kyoto_Restaurant_Info.csv")
colnames(raw.data)
```
```

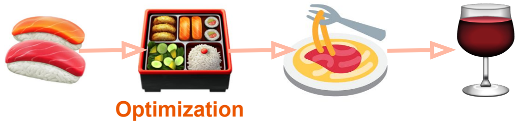
```
 [1] "X"            "Name"         "JapaneseName" "Station"      "FirstCategory" "SecondCategory"
 [7] "DinnerPrice"  "LunchPrice"   "TotalRating"  "DinnerRating" "LunchRating"   "ReviewNum"
[13] "Lat"          "Long"
```

```r
```{r}
data = read.csv("Kyoto.csv")
colnames(data)
```
```

```
 [1] "Column"                    "X"                      "Column2"
 [4] "Name"                      "JapaneseName"           "Station"
 [7] "FirstCategory"             "SecondCategory"         "DinnerPrice"
[10] "DinnerPrice.1"             "DinnerPrice.2"          "AverageDinnerPrice"
[13] "AverageDinnerPriceInUSD"   "LunchPrice"             "LunchPrice.1"
[16] "LunchPrice.2"              "AverageLunchPrice"      "AverageLunchPriceInUSD"
[19] "YenToUSD"                  "TotalRating"            "DinnerRating"
[22] "LunchRating"               "ReviewNum"              "Lat"
[25] "Long"                      "Category"
```

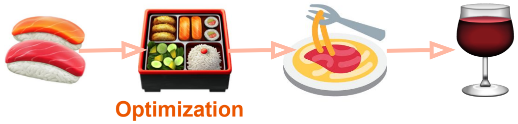
Data Structure

	Name	AverageDinnerPriceInUSD	AverageLunchPriceInUSD	DinnerRating	LunchRating	Category
1	Orudeidainingurajou	40.45	22.47	3.2	3.38	Buffet
2	Steak Frites Gaspard zinzin	31.46	13.48	3.06	3.33	Bistro
3	KAZUMA	31.46	0.0	3.28	0.0	Other
4	okonomiyakiteppanyakimiki	31.46	0.0	3.14	0.0	Okonomiyaki
5	Sumika	31.46	0.0	3.34	0.0	Fusion



Optimization of Ratings - Overview

- 3 Languages
  - Julia, w/ JuMP, GLPKSolverMIP
  - R, w/ OMPR - MIP Model
  - Python, w/ pymprog module
- Individual and Combined Models
  - Lunch model
  - Dinner model
  - Lunch + Dinner combined model



Individual Optimization Model

Objective Function:

$$MAX \frac{\sum_i^n r_i x_i}{Length\ of\ Stay}$$

Variables:

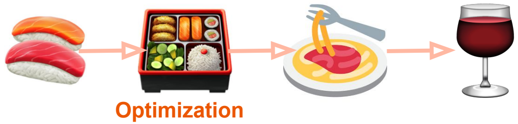
$$x_i \in \{0, 1\}$$

Constraints:

$$Maximun\ Budget \geq \sum_i^n p_i x_i \geq Minimun\ Budget$$

$$\sum_i^n x_i = Length\ of\ Stay$$

$$\sum_j^m c_j \leq 1$$



Combined Optimization Model

Objective Function:

$$MAX\ \frac{\sum_i^n \left(r_i^d x_i + r_i^l y_i\right)}{Length\ of\ Stay \times 2}$$

Variables:

$$x_i \in \{0,1\}$$

$$y_i \in \{0,1\}$$

Constraints:

$$Maximun\ Budget \geq \sum_i^n \left(p_i^d x_i + p_i^l y_i\right) \geq Minimun\ Budget$$

$$\sum_i^n x_i = Length\ of\ Stay$$

$$\sum_i^n y_i = Length\ of\ Stay$$

$$\sum_i^n (x_i + y_i) \leq 1$$

$$\sum_j^m (c_j^d + c_j^l) \leq 1$$

Optimization of Ratings - Julia

First Model: Dinner

```
using JuMP, GLPKMathProgInterface
dModel = Model(solver=GLPKSolverMIP())

preference = ["Buffet", "Bistro", "Fusion", "Bar"]
budget = [50, 100]
days = 2
df = kyoto[findin(kyoto[:Category], preference), :]
n = size(df, 1)
p = length(preference)

@variable(dModel, 0 <= x[1:n] <= 1, Int)
@constraint(dModel, sum(df[i, 2] * x[i] for i=1:n) >= minimum(budget))
@constraint(dModel, sum(df[i, 2] * x[i] for i=1:n) <= maximum(budget))
@constraint(dModel, sum(x[i] for i=1:n) == days)

for j in 1:p
    @constraint(dModel, sum((Int.(df[i,6] .== preference[[j]])) * x[i] for i=1:n) .<= 1)
end

@objective(dModel, Max, sum(df[i, 4] * x[i] for i=1:n)/days)
```

Optimization of Ratings - Julia

Suggested Restaurants - Dinner

```
dModel_solution = df[getvalue(x) .== 1,[1,2,4,6]]
```

	Name	AverageDinnerPriceInUSD	DinnerRating	Category
1	Sou	40.45	3.55	Fusion
2	burassuri-kontowa-ru	31.46	3.53	Bistro

```
println("Objective value: ", getobjectivevalue(dModel))
println("Total Dinner Price: ", "\$", sum(dModel_solution[:,2]))
```

```
Objective value: 3.54
Total Dinner Price: $71.91
```

Optimization of Ratings - Julia

Second Model: Lunch

```
using JuMP, GLPKMathProgInterface
lModel = Model(solver=GLPKSolverMIP())

preference = ["Buffet", "Bistro", "Fusion", "Bar", "Yakitori"]
budget = [70, 100]
days = 2
df = kyoto[findin(kyoto[:Category], preference), :]
n = size(df, 1)
p = length(preference)

@variable(lModel, 0 <= y[1:n] <= 1, Int)
@constraint(lModel, sum(df[i, 3] * y[i] for i=1:n) >= minimum(budget))
@constraint(lModel, sum(df[i, 3] * y[i] for i=1:n) <= maximum(budget))
@constraint(lModel, sum(y[i] for i=1:n) == days)

for j in 1:p
    @constraint(lModel, sum((Int.(df[i,6] .== preference[[j]])) * y[i] for i=1:n) .<= 1)
end

@objective(lModel, Max, sum(df[i, 5] * y[i] for i=1:n)/days)
```

Optimization of Ratings - Julia

Suggested Restaurants - Lunch

```
lModel_solution = df[getvalue(y) .== 1,[1,3,5,6]]
```

	Name	AverageLunchPriceInUSD	LunchRating	Category
1	BISTRO BAR A VIN C	31.46	3.1	Bistro
2	ORTO	40.45	3.85	Fusion

```
println("Objective value: ", getobjectivevalue(lModel))
println("Total Lunch Price: ", "\$", sum(lModel_solution[:,2]))
```

```
Objective value: 3.475
Total Lunch Price: $71.91
```

Optimization of Ratings - Julia

Third Model: Combined Model

```
using JuMP, GLPKMathProgInterface
cModel = Model(solver=GLPKSolverMIP())

preference = ["Buffet", "Bistro", "Fusion", "Bar", "Yakitori"]
budget = [120, 200]
days = 2
df = kyoto[findin(kyoto[:Category], preference), :]
n = size(df, 1)
p = length(preference)

@variable(cModel, 0 <= x[1:n] <= 1, Int)
@variable(cModel, 0 <= y[1:n] <= 1, Int)
@constraint(cModel, sum((df[i, 2] * x[i]) + (df[i, 3] * y[i]) for i=1:n) >= minimum(budget))
@constraint(cModel, sum((df[i, 2] * x[i]) + (df[i, 3] * y[i])for i=1:n) <= maximum(budget))
@constraint(cModel, sum(x[i] for i=1:n) == days)
@constraint(cModel, sum(y[i] for i=1:n) == days)

for i in 1:n
    @constraint(cModel, x[i] + y[i] <= 1)
end

for j in 1:p
    @constraint(cModel, sum(((Int.(df[i,6] .== preference[[j]])) * x[i]) +
                            ((Int.(df[i,6] .== preference[[j]])) * y[i])for i=1:n) .<= 1)
end

@objective(cModel, Max, sum((df[i, 4] * x[i]) +  (df[i, 5] * y[i]) for i=1:n)/(days*2))
```

Optimization of Ratings - Julia

Suggested Restaurants - Combined

```
dinner = df[getvalue(x) .== 1,:]
dinner[:Choice] = "Dinner"
lunch = df[getvalue(y) .== 1,:]
lunch[:Choice] = "Lunch"
cModel_solution = vcat(lunch, dinner)
```

	Name	AverageDinnerPriceInUSD	AverageLunchPriceInUSD	DinnerRating	LunchRating	Category	Choice
1	o-rudeidainingukaza	40.45	22.47	3.08	3.52	Buffet	Lunch
2	Kyounozenkuruma	40.45	13.48	3.35	3.54	Yakitori	Lunch
3	Kyoutodaina	31.46	22.47	3.53	3.32	Bistro	Dinner
4	ORTO	80.91	40.45	3.97	3.85	Fusion	Dinner

```
println("Objective value: ", getobjectivevalue(cModel))
println("Total Price: ", "\$", sum(cModel_solution[(cModel_solution[:Choice] .== "Dinner"),2])
        + sum(cModel_solution[(cModel_solution[:Choice] .== "Lunch"),3]))
```

```
Objective value: 3.6399999999999997
Total Price: $148.32
```

Optimization of Ratings - Julia

Results Comparison:

	Name	AverageDinnerPriceInUSD	DinnerRating	Category
1	Sou	40.45	3.55	Fusion
2	burassuri-kontowa-ru	31.46	3.53	Bistro

	Name	AverageLunchPriceInUSD	LunchRating	Category
1	BISTRO BAR A VIN C	31.46	3.1	Bistro
2	ORTO	40.45	3.85	Fusion

	Name	AverageDinnerPriceInUSD	AverageLunchPriceInUSD	DinnerRating	LunchRating	Category	Choice
1	o-rudeidainingukaza	40.45	22.47	3.08	3.52	Buffet	Lunch
2	Kyounozenkuruma	40.45	13.48	3.35	3.54	Yakitori	Lunch
3	Kyoutodaina	31.46	22.47	3.53	3.32	Bistro	Dinner
4	ORTO	80.91	40.45	3.97	3.85	Fusion	Dinner

```
Dinner Model:
Objective value: 3.54
Total Dinner Price: $71.91

Lunch Model:
Objective value: 3.475
Total Lunch Price: $71.91
```

```
Dinner + Lunch Model:
Average Objective value: 3.5075000000000003
Total Dinner + Lunch Price: $143.82

Combined Model:
Objective value: 3.6399999999999997
Total Price: $148.32
```

Optimization of Ratings - R

First Model - Dinner [USING OMPR - MIP Model]

```
preference = c("Buffet", "Bistro", "Fusion", "Bar")
days = 2
budget = c(50,100)
df = kyoto[kyoto[,6] %in% (preference), ]
n = nrow(df)
p = length(preference)

dModel = MIPModel() %>%
          add_variable(x[i], i = 1:n, type = "binary") %>%
          set_objective(sum_expr(df[i, 4] * x[i], i = 1:n)/days, sense = "max") %>% # rating
          add_constraint(sum_expr(df[i, 2] * x[i], i = 1:n) <= max(budget)) %>% # dinner maximumn budget
          add_constraint(sum_expr(df[i, 2] * x[i], i = 1:n) >= min(budget)) %>% # dinner minimum budget
          add_constraint(sum_expr(x[i], i = 1:n) == days)

for (j in 1:p) {
          add_constraint(dModel, sum_expr(as.numeric(df[i,6] == preference[j]) * x[i], i = 1:n) <= 1)
}
```

Optimization of Ratings - R

Suggested Restaurants - Dinner

```
(dModel_solution = df[dOpt$i,c(1,2,4,6)])
```

```
##              Name AverageDinnerPriceInUSD DinnerRating Category
## 28 Kyoutodaina                      31.46         3.53   Bistro
## 70         Sou                      40.45         3.55   Fusion
```

```
mean(dModel_solution[,3])
```

```
## [1] 3.54
```

```
sum(dModel_solution[,2])
```

```
## [1] 71.91
```

Objective value: 3.54 [Average Rating]

Total Dinner Price: $ 71.91

Optimization of Ratings - R

Suggested Restaurants - Dinner

```
(dModel_solution = df[dOpt$i,c(1,2,4,6)])
```

```
##               Name AverageDinnerPriceInUSD DinnerRating Category
## 28 Kyoutodaina                     31.46         3.53   Bistro
## 70         Sou                     40.45         3.55   Fusion
```

```
mean(dModel_solution[,3])
```

```
## [1] 3.54
```

```
sum(dModel_solution[,2])
```

```
## [1] 71.91
```

Julia

	Name	AverageDinnerPriceInUSD	DinnerRating	Category
1	Sou	40.45	3.55	Fusion
2	burassuri-kontowa-ru	31.46	3.53	Bistro

Objective value: 3.54 [Average Rating]

Total Dinner Price: $ 71.91

Optimization of Ratings - R

Second Model - Lunch [USING OMPR - MIP Model]

```r
preference = c("Buffet", "Bistro", "Fusion", "Bar", "Yakitori")
days = 2
budget = c(70,100)
df = kyoto[kyoto[,6] %in% (preference), ]
n = nrow(df)
p = length(preference)

lModel = MIPModel() %>%
          add_variable(x[i], i = 1:n, type = "binary") %>%
          set_objective(sum_expr(df[i, 5] * x[i], i = 1:n)/days, sense = "max") %>% # rating
          add_constraint(sum_expr(df[i, 3] * x[i], i = 1:n) <= max(budget)) %>% # lunch price
          add_constraint(sum_expr(df[i, 3] * x[i], i = 1:n) >= min(budget)) %>%
          add_constraint(sum_expr(x[i], i = 1:n) == days)
for (j in 1:p) {
          add_constraint(lModel, sum_expr(as.numeric(df[i,6] == preference[j]) * x[i], i = 1:n) <= 1)
}
```

Optimization of Ratings - R

Suggested Restaurants - Lunch

```
(lModel_solution = df[lOpt$i,c(1,3,5,6)])
```

```
##                      Name AverageLunchPriceInUSD LunchRating Category
## 155 BISTRO BAR A VIN  C                   31.46        3.10   Bistro
## 609                  ORTO                   40.45        3.85   Fusion
```

```
mean(lModel_solution[,3])
```

```
## [1] 3.475
```

```
sum(lModel_solution[,2])
```

```
## [1] 71.91
```

Objective value: 3.475 [Average Rating]

Total Lunch Price: $ 71.91

Optimization of Ratings - R

Suggested Restaurants - Lunch

```
(lModel_solution = df[lOpt$i,c(1,3,5,6)])
```

```
##                    Name AverageLunchPriceInUSD LunchRating Category
## 155 BISTRO BAR A VIN  C                  31.46        3.10   Bistro
## 609                ORTO                  40.45        3.85   Fusion
```

```
mean(lModel_solution[,3])
```

```
## [1] 3.475
```

```
sum(lModel_solution[,2])
```

```
## [1] 71.91
```

Julia

	Name	AverageLunchPriceInUSD	LunchRating	Category
1	BISTRO BAR A VIN C	31.46	3.1	Bistro
2	ORTO	40.45	3.85	Fusion

Objective value: 3.475 [Average Rating]

Total Lunch Price: $ 71.91

Optimization of Ratings - R

Third Model - Combined Model [USING OMPR - MIP Model]

```
preference = c("Buffet", "Bistro", "Fusion", "Bar", "Yakitori")
days = 2
budget = c(120,200)
df = kyoto[kyoto[,6] %in% (preference), ]
n = nrow(df)
p = length(preference)

cModel = MIPModel() %>%
          add_variable(x[i], i = 1:n, type = "binary") %>% # x = dinner
          add_variable(y[i], i = 1:n, type = "binary") %>% # y = lunch
          set_objective(sum_expr((df[i, 4] * x[i]) + (df[i, 5] * y[i]), i = 1:n) / (2 * days), sense = "max") %>%
          add_constraint(sum_expr((df[i, 2] * x[i]) + (df[i, 3] * y[i]), i = 1:n) <= max(budget)) %>%
          add_constraint(sum_expr((df[i, 2] * x[i]) + (df[i, 3] * y[i]), i = 1:n) >= min(budget)) %>%
          add_constraint(sum_expr(x[i], i = 1:n) == days) %>%
          add_constraint(sum_expr(y[i], i = 1:n) == days) %>%
          add_constraint((x[i] + y[i]) <= 1, i = 1:n)

for (j in 1:p) {
          add_constraint(cModel, sum_expr((as.numeric(df[i,6] == preference[j]) * x[i]) + (as.numeric(df[i,6] == preference
[j]) * y[i]), i = 1:n) <= 1)
}
```

Optimization of Ratings - R

Suggested Restaurants - Combined

```
                                Name AverageDinnerPriceInUSD AverageLunchPriceInUSD DinnerRating LunchRating Category Choice
70                               Sou                   40.45                   0.00         3.55        0.00   Fusion Dinner
609                             ORTO                   80.91                  40.45         3.97        3.85   Fusion Dinner
188 pasutakorekushonandoba-dougetsuneo                 40.45                  13.48         3.09        3.52   Bistro  Lunch
410                     Kyounozenkuruma                40.45                  13.48         3.35        3.54 Yakitori  Lunch
```

```
(sum(cModel_solution[,4][cModel_solution[,7] == "Dinner"]) + sum(cModel_solution[,5][cModel_solution[,7] == "Lunch"]))/nrow
(cModel_solution)
```

```
## [1] 3.645
```

```
(sum(cModel_solution[,2][cModel_solution[,7] == "Dinner"]) + sum(cModel_solution[,3][cModel_solution[,7] == "Lunch"]))
```

```
## [1] 148.32
```

Combined Model:

Objective value: 3.645

Total Price: $148.32

Optimization of Ratings - R

Suggested Restaurants - Combined

	Name	AverageDinnerPriceInUSD	AverageLunchPriceInUSD	DinnerRating	LunchRating	Category	Choice
70	Sou	40.45	0.00	3.55	0.00	Fusion	Dinner
609	ORTO	80.91	40.45	3.97	3.85	Fusion	Dinner
188	pasutakorekushonandoba-dougetsuneo	40.45	13.48	3.09	3.52	Bistro	Lunch
410	Kyounozenkuruma	40.45	13.48	3.35	3.54	Yakitori	Lunch

Julia

```
Combined Model:
Objective value: 3.6399999999999997
Total Price: $148.32
```

Combined Model:

Objective value: 3.645

Total Price: $148.32

	Name	AverageDinnerPriceInUSD	AverageLunchPriceInUSD	DinnerRating	LunchRating	Category	Choice
1	o-rudeidainingukaza	40.45	22.47	3.08	3.52	Buffet	Lunch
2	Kyounozenkuruma	40.45	13.48	3.35	3.54	Yakitori	Lunch
3	Kyoutodaina	31.46	22.47	3.53	3.32	Bistro	Dinner
4	ORTO	80.91	40.45	3.97	3.85	Fusion	Dinner

Optimization of Ratings - R

Results Comparison:

Dinner Model:

Objective value: 3.54

Total Dinner Price: $71.91

Lunch Model:

Objective value: 3.475

Total Lunch Price: $71.91

Dinner + Lunch Model:

Average Objective value: 3.5075

Total Dinner + Lunch Price: $143.82

Combined Model:

Objective value: 3.645

Total Price: $148.32

Optimization of Ratings - Python

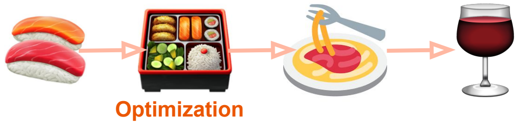
Using pymprog package ⇒ Dinner Model

```python
from pymprog import *
import numpy as np

preference = ("Buffet", "Bistro", "Fusion", "Bar", "Yakitori")
days = 3
budget = [50, 100]
df = new_kyoto[(new_kyoto['Category'] == 'Buffet') | (new_kyoto['Category'] == "Bistro") |
          (new_kyoto['Category'] == 'Fusion') | (new_kyoto['Category'] == "Bar")]
n = len(df)
A= list(new_kyoto['DinnerRating'])
c = list(new_kyoto['AverageDinnerPriceInUSD'])

begin('dinner') # begin modelling
verbose(True)
x = var('x',n, kind = bool) # set up the variiable
rating = sum(A[i]*x[i] for i in range(n))/days # init
maximize(rating) # define the direction
#  Set constraints
R1 = sum(p*q for p,q in zip(c,x)) >= min(budget)
R2 = sum(p*q for p,q in zip(c,x)) <= max(budget)
R3 = sum(x[i] for i in range(n)) == days
solve()
end()
```

```
GLPK Simplex Optimizer, v4.61
3 rows, 58 columns, 174 non-zeros
      0: obj =  -0.000000000e+00 inf =   5.300e+01 (2)
      5: obj =   3.123831271e+00 inf =   0.000e+00 (0)
*    11: obj =   3.538255835e+00 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
GLPK Integer Optimizer, v4.61
3 rows, 58 columns, 174 non-zeros
58 integer variables, all of which are binary
Integer optimization begins...
+    11: mip =     not found yet <=              +inf        (1; 0)
Solution found by heuristic: 3.53333333333
+    13: mip =   3.533333333e+00 <=     tree is empty   0.0% (0; 1)
INTEGER OPTIMAL SOLUTION FOUND
```



Challenges

- Struggled with the objective function ⇒ diverse dataset, lots of room for experimentation
- Translating the goal into a set of mathematical equations
- Tried running multiple models in three programming languages
- Experimented with various optimization packages

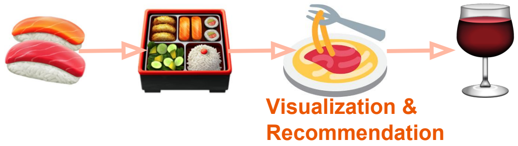
Visualization & Recommendation

Visualization and Recommendation

Days	Budget	Cuisine	Restaurant	TotalCost	Category	Rating
1	15, 30	"Buffet", "Bistro", "Fusion", "Bar", "Seafood", "Local"	BISTRO BAR A VIN C	$ 22.47	Bistro	3.51
1	30, 50	"Buffet", "Bistro", "Fusion", "Bar", "Seafood", "Local"	Baruagiyao	$ 49.44	Seafood	3.57
2	30, 60	"Buffet", "Bistro", "Fusion", "Bar", "Seafood", "Local", "Kappo", "Yakini	Kyoutodaina, Tsushima	$ 44.94	Bistro, Seafood	3.5
2	40, 80	"Shabu Shabu", "Bistro", "Fusion", "Bar", "Seafood", "Local", "Kappo", "	Ishibekoujimamecha, Kyoutodaina	$ 71.91	Bistro, Local	3.545
2	50, 100	"Shabu Shabu", "Bistro", "Fusion", "Bar", "Seafood", "Local", "Kappo", "	Tsushima, ORTO	$ 71.91	Fusion, Seafood	3.72
3	40, 60	"Shabu Shabu", "Sumibiyaki", "Horumon", "Ramen", "Seafood", "Local"	Ishibekoujimamecha, Tsushima, chuukasobatakayasu	$ 58.42	Local, Seafood, Ramen	3.54
3	40, 80	"Shabu Shabu", "Sumibiyaki", "Horumon", "Ramen", "Seafood", "Local"	Ishibekoujimamecha, gyokouchokusoukaisensakabaanji, chuukasoba	$ 76.39	Local, Seafood, Ramen	3.56
3	60, 120	"Shabu Shabu", "Sumibiyaki", "Horumon", "Ramen", "Seafood", "Local"	Aritsune, gyokouchokusoukaisensakabaanji, chuukasobatakayasu	$ 116.86	Kappo, Seafood, Ramen	3.59

Highlights from the Dinner Model, Sample table:

- Cuisine preference considered
- Estimate costs within budget planning
- Rating maximized

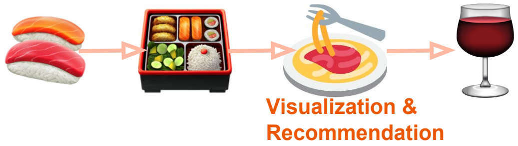


Visualization and Recommendation

Days	Budget	Cuisine	Lunch Restaurant	Dinner Restaurant	Category	Rating
1	20, 120	"Okonomiyaki", "Sukiyaki", "Buffet", "Ramen", "Sumibiyaki", "Tonkatsu", "Mizutaki"	chuukasobatakayasu	Takasegawa	Ramen, Sumibiyaki	3.595
1	40, 240	"Cafe", "Hot Pot", "Steak", "Wine Bar", "Bistro"	ORENO PAN okumura	Puranchaken	Café, Steak	3.545
2	20, 120	"Okonomiyaki", "Sukiyaki", "Buffet", "Ramen", "Sumibiyaki", "Tonkatsu", "Mizutaki"	chuukasobatakayasu, Menshoutakamatsu	Yumeya, Takasegawa	Ramen, Sumibiyaki, Okonomiyaki	3.58
2	80, 240	"Okonomiyaki", "Sukiyaki", "Buffet", "Ramen", "Sumibiyaki", "Tonkatsu", "Mizutaki"	o-rudeidainingukaza, Menshoutakamatsu	chuukasobatakayasu, Takasegawa	Buffet, Ramen, Sumibiyaki	3.57
3	60, 120	"Fusion", "Dining Bar", "Steak", "Yakiniku", "Seafood"	ANAGOYA NORESORE, sumibisute-kisakaikyoutosanjou, Kamogawatakashi	Sou, Baruagiyao, ORTO	Fusion, Seafood, Steak, Yakiniku	3.63

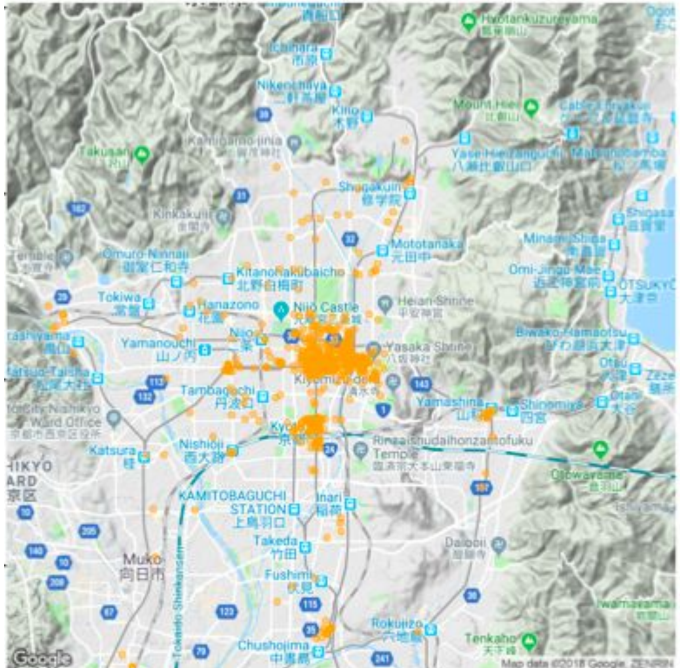
Highlights from the Combined Model, Sample table:

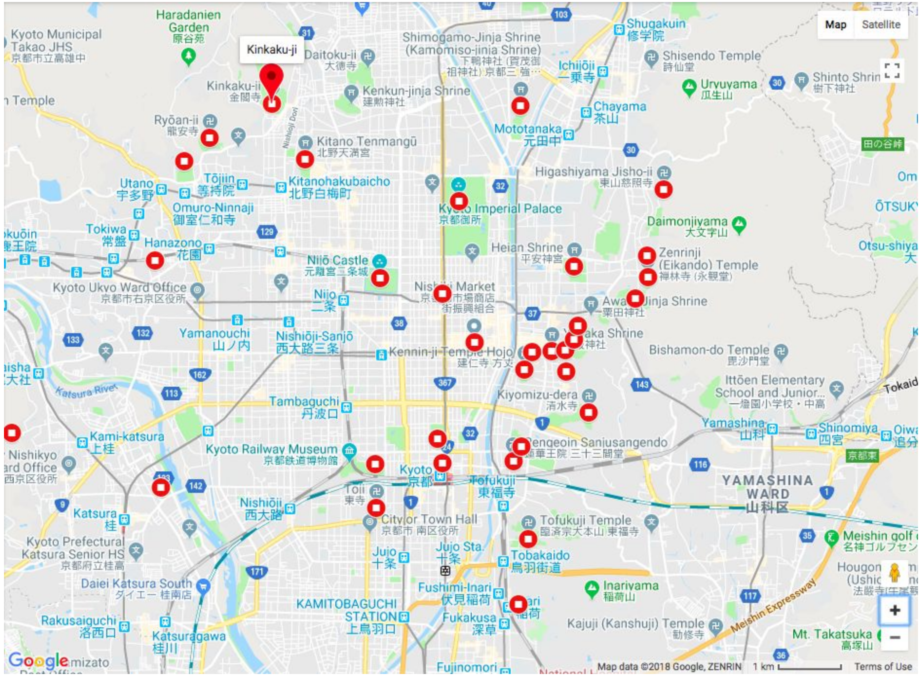
- Cuisine preference considered
- Different restaurants recommended w/o repetition
- Everything within the preference pre-set, without too many repetitions



Visualization on Maps

Double check our projections:





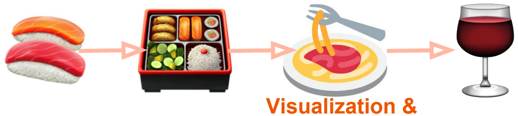
Source: Things to do in Kyoto, from Google Maps

Visualization on Maps

“Passerby Economy”

- Quick 2-day visit
- Budget-friendly ($50 ~ 80)
- Lunch recommendations only
- Categories of interest:
    - "Buffet", "Bistro", "Fusion", "Bar", "Yakitori"
- Maximum rating: **3.695**
- Where to eat:

Name <fctr>	AverageLunchPriceInUSD <dbl>	LunchRating <dbl>	Category <fctr>
Kyounozenkuruma	13.48	3.54	Yakitori
ORTO	40.45	3.85	Fusion

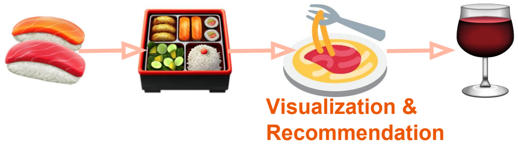


Visualization on Maps

“Fancy Traveler”

- Deep-dive 5-day stay
- Indulge yourself ($200 ~ 500)
- Dinner recommendations only
- Categories of interest:
    - "Okonomiyaki", "Sukiyaki", "Bar", "Horumon", "Sumibiyaki", "Tonkatsu", "Mizutaki", "Kushiyaki"
- Maximum rating: **3.414**
- Where to eat:

Name <fctr>	AverageDinnerPriceInUSD <dbl>	DinnerRating <dbl>	Category <fctr>
torisemmontenyamadori	31.46	3.38	Sukiyaki
kaitoshirowainnobarukakimaru	31.46	3.31	Bar
L'ajitto	80.91	3.23	Okonomiyaki
Yumeya	13.48	3.55	Okonomiyaki
Takasegawa	49.44	3.60	Sumibiyaki

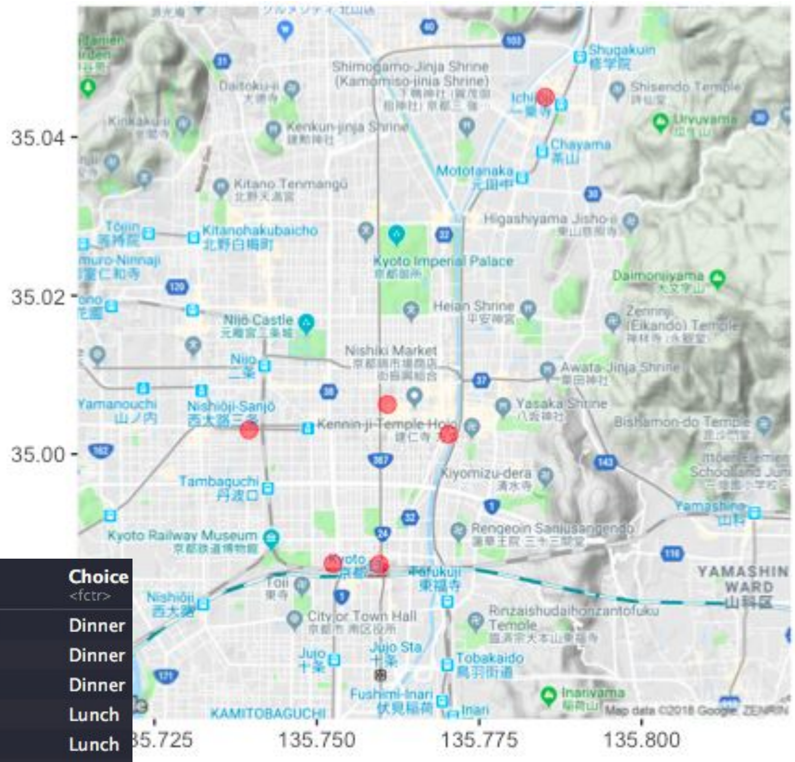


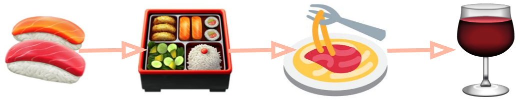
Visualization on Maps

“Family Economy”

- Family friendly, 3-day stay
- Budget between $120 ~ 180
- Categories of interest:
  - "Okonomiyaki", "Sukiyaki", "Buffet", "Ramen", "Sumibiyaki", "Tonkatsu", "Mizutaki"
- Maximum rating: **3.55**
- Where to eat:

Name <fctr>	DinnerRating <dbl>	LunchRating <dbl>	Category <fctr>	Choice <fctr>
Mossannobetayaki	3.51	3.06	Okonomiyaki	Dinner
chuukasobatakayasu	3.58	3.59	Ramen	Dinner
Takasegawa	3.60	0.00	Sumibiyaki	Dinner
o-rudeidainingukaza	3.08	3.52	Buffet	Lunch
Cafe Restaurant Le Temps	3.26	3.50	Buffet	Lunch
Menshoutakamatsu	3.41	3.58	Ramen	Lunch



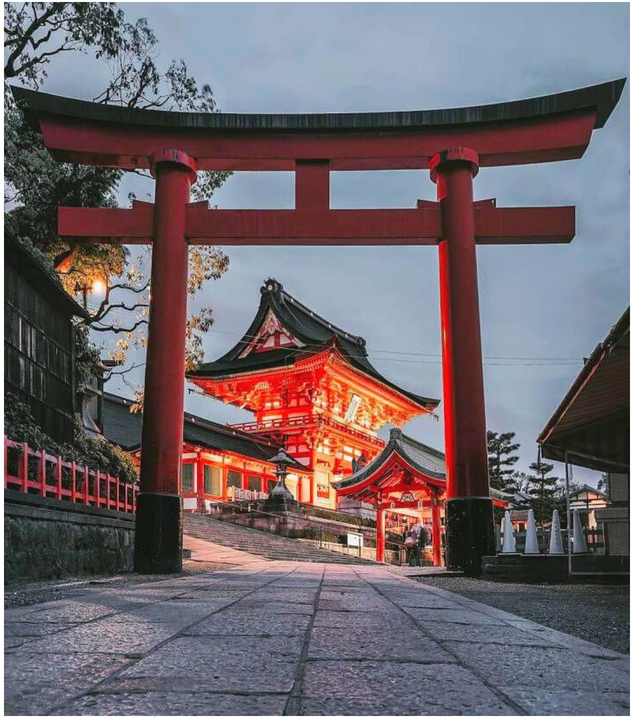


Conclusion & Future Work

Conclusion and Future Work

- Overall results we got
- Three models we obtained
  - highly scalable
  - not dependent on any particular scale of the rating
  - highly customizable

- Potential addition of the nutrition information and dietary restrictions
- Interactive dashboard
- Mobile App for iOS and Android



Happy traveling! ✈️